CLAIMS

What is claimed is:

1. A driver management system, comprising:

a driver framework component (DFC) that is separate from a driver, the DFC comprising:

a presentation component that selectively exposes adapter objects to the driver in a multi-threaded environment.

2. The system of claim 1, further comprising an operating system kernel that operates or performs in a multi-threaded software environment.

3. The system of claim 2, the driver of claim 1 operates in a less-threaded software environment than the operating system kernel.

4. The system of claim 1, the adapter object includes internal state data and one or more sets of locks for managing interactions between the driver and the DFC.

5. The system of claim 4, the adapter object includes an internal object state lock that provides synchronization for modifications to the internal state data.

6. The system of claim 5, the internal object state lock is acquired and released for short time intervals in response to an event from a system that modifies the adapter object, or an API call from a software module or driver.

7. The system of claim 4, the adapter object includes a presentation lock that is acquired when events are presented through event handler callbacks into a less threaded software module.

8. The system of claim 7, when the event handler callback returns, the presentation lock is automatically released.

9. The system of claim 1, the object adapter employs a series of reference counts, request deferrals, or other programming components to facilitate object lifetime and event exposure to a less threaded software module.

10. The system of claim 1, the adapter objects are employed for request dispatch, locking, or synchronization.

11. The system of claim 1, the DFC automatically manages synchronization and race conditions that occur in a driver environment.

12. The system of claim 11, the DFC provides a flexible configuration model in which a driver designer can select an amount of synchronization desired depending on device requirements or performance goals.

13. The system of claim 1, the driver registers a set of callback functions to the adapter objects during initialization of the driver.

14. The system of claim 1, the DFC raises events that occur such as Delayed Procedure Calls (DPC's), I/O cancellation events, plug and play events, or power management events.

15. The system of claim 1, the adapter object is associated with at least one of a request object, a driver object, a device object, and a queue object.

16. The system of claim 15, at least one of the objects is owned or derived from at least one other object.

17. The system of claim 1, the adapter object including at least one of a spinlock, a shared lock, and a FAST_MUTEX.

18. The system of claim 1, further comprising a tuning component to automatically adjust performance over time as the DFC is profiled.

19. The system of claim 1, Fig. 4 the adapter object allows the driver to specify an optional Context memory allocation to be associated with a handle.

20. The system of claim 1, the adapter object is associated with a hierarchical locking model.

21. The system of claim 20, the hierarchical locking model includes at least one of a per driver model, a per device model, a per queue model, and a non-synchronization model.

22. The system of claim 1, further comprising at least one of a synchronous and an asynchronous threading model.

23. The system of claim 1, further comprising at least one of a lock for inter-object communications, a lock verifier, a lock organizer, and an automatic child-locking component.

24. A computer readable medium having computer readable instructions stored thereon for implementing the DFC and the presentation component of claim 1.

25. A computer-based driver interface system, comprising:

means for performing highly threaded software operations;

means for performing lower threaded software operations;

means for interfacing between the highly threaded software operations and the lower threaded software operations; and

means for automatically managing events between the highly threaded software operations and the lower threaded software operations.

26. The system of claim 25, further comprising means for serializing the events.

27. A method to facilitate driver interactions in accordance with an operating system, comprising:

adapting an object with one or more locks;

presenting the object to a driver *via* an interface framework; and

employing the object to interface between a system having at least one more thread than the driver.

28. The method of claim 27, further comprising automatically serializing events between the driver and the system.

29. The method of claim 27, the one or more locks further comprising at least one of an internal state lock and a presentation lock.

30. The method of claim 27, further comprising locking events between objects.

31. The method of claim 27, further comprising controlling the object lifetime.

32. The method of claim 27, further comprising hierarchically controlling the locks.

33. A computer readable medium having a data structure stored thereon, comprising:

a first data field related to a framework object having a handle associated therewith to facilitate data access;

a second data field that relates to a driver component that communicates *via* the handle and receives events from the framework object; and

a third data field that presents one or more locks for the framework object.

34. The computer readable medium of claim 33, further comprising at least one Application Programming Interface (API).